PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 :
G06F 15/21, 17/30, 19/00

A1

(11) International Publication Number: WO 00/43901

(43) International Publication Date: 27 July 2000 (27.07.00)

(21) International Application Number: PCT/US00/00737

(22) International Filing Date: 12 January 2000 (12.01.00)

(30) Priority Data:
09/234,551 20 January 1999 (20.01.99) US

(71) Applicant: CHANNELPOINT, INC. [US/US]; 10155 Westmoor Drive, Suite 210, Westminster, CO 80020 (US).

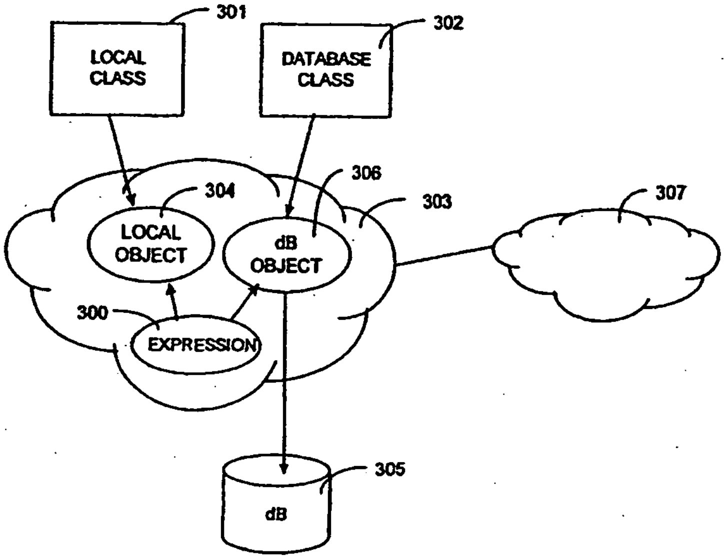
(72) Inventor: JACKSON, Jerry, R.; 11270 Wakely Road, Colorado Springs, CO 80908 (US).

(74) Agents: BURTON, Carol, W. et al.; Hogan & Hartson LLP, Suite 1500, 1200 17th Street, Denver, CO 80202 (US).

(81) Designated States: AU, CA, JP, NZ, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: RATING ENGINE CONSTRAINT PROCESSING



(57) Abstract

A computer (111) implemented rating methodology for generating rate quotes in a plurality of externally defined contexts. A database (305) includes records describing objects to be rated. At least one context–specific constraint module (203) is defined that is selectively enabled and selectively called to constrain rating of selected records. A context–generic rating module (201) is provided comprising programming constructs that access records from the dabase (305), perform basic computations on the records to generate output data, and selectively enable the constraint module (203) and call constraints defined in the rate module (201).

AQ2

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece	ML	Mali	TR	Turkey
BG	Bulgaria	HU	Hungary	MN	Mongolia	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MR	Mauritania	UA	Ukraine
BR	Brazil	IL	Israel	MW	Malawi	UG	Uganda
BY	Belarus	IS	Iceland	MX	Mexico	US	United States of America
CA	Canada	IT	Italy	NE	Niger	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NL	Netherlands	VN	Viet Nam
CG	Congo	KE	Kenya	NO	Norway	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NZ	New Zealand	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	PL	Poland		
CM	Cameroon	KR	Republic of Korea	PT	Portugal		
CN	China	KZ	Kazakstan	RO	Romania		
CU	Cuba	LC	Saint Lucia	RU	Russian Federation		
CZ	Czech Republic	LI	Liechtenstein	SD	Sudan		
DE	Germany	LK	Sri Lanka	SE	Sweden		
DK	Denmark	LR	Liberia	SG	Singapore		
EE	Estonia						

RATING ENGINE CONSTRAINT PROCESSING

BACKGROUND OF THE INVENTION

1. Field of the Invention.

The present invention relates, in general, to database software and computer program products, and, more particularly, to database software for implementing rating methodologies for insurance industry applications.

2. Relevant Background.

The present invention involves methods, systems, and computer program products used to create rating methodologies for automated underwriting systems. A rating methodology is a collection of business rules that enable an insurance provider to provide a rate or cost of a particular product. Insurance products are unique in that the rate for a product must take into consideration market considerations, geographic considerations, actuarial information, and government regulatory information (among other considerations). These often competing considerations must be balanced and continuously updated to provide a competitive and profitable product.

A rating methodology is typically developed by actuaries and business analysts who understand the insurance industry and customer needs. Typically the methodology is expressed in domain specific terms and expressions that can be communicated easily between the analysts and actuaries. However, these domain specific terms and expressions do not readily translate into computer readable program code. Hence, computer programmers translate the rating methodology into a software implementation. This translation process is costly, error prone, and time consuming. Analysts who designed the original methodology cannot independently verify that the software translation is an accurate representation of the methodology. Moreover, the resulting software often contains machine specific program code that is not portable between mainframes, workstations, and personal computers.

The insurance industry is an information intensive industry that relies heavily on information technologies. The insurance industry undergoes rapid change that is reflected in rapidly changing demands on those information technologies. For example, an insurance company may expand into new geographic markets or combine its product portfolio with other business entities through mergers and acquisitions. As a result, the information systems that implement the companies rating systems are continuously adjusted and modified to account for new product types, new markets, and new regulatory environments. Further, because the insurance industry in such a heavily regulated business, a company's systems must adapt to new regulations and legislation simply to stay in business in an existing market. As a result, it is crucial to have flexible, portable rating system that readily adapts to changes in user environments and regulatory environments in which the insurance carrier operates.

COBOL is widely used for rating applications because none of the languages that have become popular in the last three decades aid in overcoming the limitations set out above. Most of the advances embodied in popular programming languages since COBOL (e.g., BASIC, FORTRAN, C, C++, and JAVA) offer improvements to COBOL that are simply irrelevant to common business applications such as insurance rating that function essentially to transform database inputs into database outputs. Principle functionality desired in these applications includes:

- Simplified database access integrated into the language;
- Support for direct manipulation of sets of records without complex loops, arrays, and the like;
- Runtime configuration based on business logic and constraints;
- Rule-based deduction;

- Automatic generation of user interface components; and
- Portability across all levels of enterprise computing.

Conceptually, many limitations of the prior art result because the problem to be solved, i.e., implementing a rating methodology, is merged with the programming logic that is used to implement the methodology. Because of this merger, program development and testing increase in complexity dramatically. Small changes in the methodology due to expanded product portfolio or legislative changes required significant programming effort to implement.

Similarly, porting an existing rating methodology to a new computer system required a similar level of programming effort. Hence, it becomes prohibitive to take advantage of new hardware and operating environments. As a result, many existing rating methodology systems remain on older mainframe computer systems implemented in COBOL code that is bulky and difficult to maintain. A need exists for implementing methodologies that separates the physical and environmental factors that dictate the description of the methodology from the program code used to implement the methodology on a computer system.

SUMMARY OF THE INVENTION

Briefly stated, the present invention involves a computer implemented rating methodology for generating rate quotes in a plurality of externally defined contexts. A database includes records describing objects to be rated. At least one context-specific constraint module is defined that is selectively enabled and selectively called to constrain rating of selected records. A context-generic rating module is provided comprising programming constructs that access records from the database, perform basic computations on the records to generate output data, and selectively enable the constraint module and call constraints defined in the rate module.

In another aspect, the present invention involves a computer system for implementing a rating methodology to generate rate quotes for a set of data objects. A first computer having a processor and memory is coupled to a database comprising records describing the set of data objects to be rated. A plurality of context-specific constraint modules are selectively enabled to constrain rating of selected data objects. A context-generic rating module comprising programming constructs that access records from the database performs basic computations on the records to generate output data, and to selectively enable the constraint module.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 shows a networked computer environment implementing the system, method and devices in accordance with the present invention;

FIG. 2 illustrates basic program devices in accordance with an embodiment of the present invention;

FIG. 3 illustrates in block diagram for interaction of program devices to implement a method in accordance with the present invention;

FIG. 4 shows a flow diagram illustrating basic steps in accordance with a compilation process in accordance with the present invention;

FIG. 5 shows a flow diagram illustrating basic steps in accordance with a runtime execution of the computer implemented devices and method in accordance with the present invention; and

FIG. 6 shows exemplary data structures useful in the implementation of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

FIG. 1 illustrates a typical distributed computing environment in which the present invention may be implemented. In overview, FIG. 1 shows general and/or special purpose computers, workstations or personal computers that are connected via communications links of various types. Programs and data, many in the form of objects, are made available by various members of the system for execution and access by other members of the system.

The representative computer system shown in FIG. 1 includes a workstation or personal computer (PC) 111 and associated server 101 coupled together through an appropriate communications link. The workstation 101 may include input/output ("I/O"), central processing unit ("CPU") and memory sections (not shown) and an associated monitor for interacting with a user. A variety of input devices, such as a mouse or keyboard, form a portion of the workstation 101 and are coupled to the I/O section to provide user input data.

Workstations 111 typically includes mass storage devices such as CDROM and hard disk devices (not shown) for read only and read-write storage. Additionally, workstation 111 may access external mass storage devices such as disk array 102 that is directly connected to server 101 and disk array 103 and tape storage 104 that are coupled through network or fiber 116. Network 116 may be implemented as a wide area network (WAN), local area network (LAN) and may use any available technology for establishing communication links such as Ethernet, Fibre Channel (FC), Internet Protocol (IP), asynchronous transfer mode (ATM), digital subscriber line (DSL), and the like. Network 116 may also couple to external LAN or WAN subnetworks such as LAN 108 including workstations 112 and 113 and a server 110 coupled together by a hub 109.

The computer program products containing mechanisms to effectuate the apparatus and methods of the present invention may reside in the memory portions of the workstations 111, 112 and 113 as well as servers 101 and 110 or any of the various associated computer mass storage devices such as tape drive 104, disk arrays 102 and 103. The computer program products containing mechanisms to effectuate the apparatus and methods of the present invention are readily embodied in magnetic, optical, magneto-optical or other available machine readable encoding systems.

The present invention is described in terms of a new computer language called PROBOL, although the teachings of the invention can be applied an implemented in a number of other programming environments including JAVA programming environment. JAVA is a trademark of Sun Microsystems, Inc., Palo Alto, California. The present invention is desirably implemented using modular program components as shown in FIG. 2. Modular components can be reused and are easier to maintain. Updates can be made to only one place in the code, and problems usually have only one source.

Modules are simply chunks of code into which an entire program is divided for manageability. Modules comprise a set of statements, preferably declarative statements, that describe the program constructs that that module uses to implement a programatically defined behavior. A module is activated explicitly or implicitly only as needed. Hence, modules that are not used will not consume computational resources other than the mass storage used to archive the module code. Because only the modules that are needed are loaded and executed, any particular execution of the rating methodology can be accomplished with a small, portable amount of code. Moreover, modules can be maintained, upgraded, added and deleted in a manner that will effect only rating methodology implementations that reference that module.

The present invention is usefully understood in terms of a program 200 comprising three particular types of modules as shown in FIG. 2. A rate module 201 comprises the main definition that references the other modules and directs the execution of the entire program. Rate module 201 includes methods and computer program product devices for implementing a set of calculations that calculate a rate from a set of input data. Constraint modules 203 comprise definitions of filters, modifiers, and value adjustments used to adapt a general-purpose methodology to different legislative constraints. Library modules 202 contain definitions of classes and other supporting procedures and functions that can be referenced by a rate or constraint module. It should be understood that this classification and naming of module types is somewhat arbitrary and more or fewer module types may defined in a particular application. The specific classifications presented herein serve as illustration and not limitation of the present invention.

In the particular example, a rating methodology is implemented as a software program 200 that must at least have a rate module 201. Typically it will have several library modules 202 and constraint modules 203. Alternatively, all program constructs can be defined in the rate module 201, although this practice is cumbersome and inefficient for a program 200 of significant size. Constraints, however are defined in a constraint module 203 in accordance with the present invention so that a constraint used in a particular application does not encumber all other rating methodologies that use that same rate module 201.

The present invention particularly involves a class of constructs within the PROBOL programming environment that implement constraints in a constraint module 203. Constraints comprise constructs that operate on a set of input data instances (e.g., a set of instances of a database class or a set of instances of calculated rate quotes). Additionally, some types of constraints (e.g., value adjustments discussed hereinbelow) may operate on primitive values as well. In general, constraints generate an output set of data instances that varies from the input set in a manner defined by the constraint definition. In the particular examples given herein, constraints come in three varieties: filters, modifiers, and adjustments. Referring to FIG. 3, the program devices that implement a calculation are illustrated as a cloud 303 that includes, for example, a "main" definition and one or more procedure definitions that are called by the main definition. The main definition is provided in the a rate module 201 shown in FIG. 2. The procedure definitions may be provided in either the rate module or by reference to procedures stored in library module 202.

Main definitions provide several commands, such as SAVE, MODIFY, and DELETE, that direct the way database class 301 interacts with its associated database table in database 305. Other commands in main definition control the flow of command execution. Each program has only one main definition that provides centralized control for the program's database operations. Procedures provide capabilities similar to the main definitions, but can be used throughout the program. Procedures must eventually be called directly or indirectly from a main definition, they give you a way to distribute program control, making the main definition more compact.

Database class 301 comprises a data structure that describes table data that can be used in a calculation. Database class 301 is associated with a particular database table and generates a set of input objects from the associated database table. As a result, when you change information in a database class, the change is also made to its associated database table in database 305.

Filter 302 operates to exclude inappropriate or undesired data from a calculation. Filters serve to match insurance plans and products with groups of prospective buyers with a minimum of programming effort. When a rate is calculated by a rate module 201, one or more filters is operative to exclude from the calculation all the data that does not meet the criteria defined for a particular plan or product. In this manner, one general purpose rate calculation implemented by a rate module 201 can be used for several different products. Because filters 302 exclude class instances that do not apply to a particular product, the filter 302 customizes a general purpose calculation to the needs of a particular product.

Every filter 302 is associated with a particular database class. Once active and called, a filter 302 examines instances of its associated database class. A class, however, may have more than one associated filter. In a particular implementation, the definition of a filter 302 includes a condition list that determines which instances the filter returns. By way of example, a condition list for a filter definition may include an age condition on a particular field of the database instance (e.g., dependents less than 18 years old), a residence condition (e.g., all dependents must have the same address) or an occupation condition (e.g., primary insured must be a union member). Only the instances that satisfy the conditions in the condition list are returned when a filter is called.

Modifiers 304 are similar to filters 302 but differ in that they alter a rate calculation after it is computed. Like filters 302, modifiers 304 include in their definition a condition list that determines whether the modifier is applied to a particular class instance. Modifiers are preferably implicitly called by expressions in the rate module 201 that save class instances to database 305. Although modifications are implicitly called, object modifications can be performed explicitly. For example, given a modifier like:

```
  MODIFIER table:xyz x IS
    STARTSWITH(x.name, "a")
    ASSIGN(size ← x.size + 1,
         name ← SUBSTRING(x.name, 1, LENGTH(x.name)))
END MODIFIER
```

there is no way to invoke it explicitly though you could explicitly perform the steps:

```
MODIFY x ASSIGN(size ← x.size + 1,
   name ← SUBSTRING(x.name, 1, LENGTH(x.name)))
```

which would produce the same change in the object 'x'.

By way of example, a condition list for a modifier definition may include a legislatively imposed rate limit (e.g., rate must be less than $25/month) or an imposed range (e.g., rate must fall within 90%-110% of a legislatively imposed standard rate). Whereas filters are applied to instances that are entering a rate module 201, modifiers are applied to instances that are leaving a rate module 201. Filters exclude instances that do not satisfy the condition list specified in the filter definition whereas modifiers operate to modify instances that meet a specified condition and pass through unmodified those instances that do not meet the specified condition.. Modifiers are associated with a particular database class and examine only instances of the associated class. As with filters, a particular database class may be associated with multiple modifiers 304. In operation, the modification, if applied, changes the value of the instance about to be saved back to the database 305.

Value adjustments 306, like filters 302 and modifiers 304, are constraints that are defined in a constraint module. Value adjustments enable one general purpose calculation implemented in a rate module 201 to be automatically customized under user specified conditions. Unlike filters and modifiers, however, value adjustments 306 can be used one or more times throughout the calculation performed by a rate module 201, not just on class instances heading in or out of the rate module 201.

In FIG. 3, cloud 303 represents the portions of main module 201 that implement a rate calculation. In the particular implementation, a value adjustment that is defined in an attached (i.e., active) constraint module 203 is called by an "ADJUST" expression in the rate module 201. The ADJUST expression includes an argument identifying the name or ID of the particular value adjustment 306 that is to be applied. In a preferred implementation, each adjustment is defined for a particular data type (e.g., float, integer, and the like). A value adjustment 306 will not be applied even when called if the type does not match.

Value adjustments 306 operate to alter a computed rate after it has been calculated so that it falls within certain parameters, usually to meet guidelines established by regulatory agencies or legislation. In the example shown below the value adjustment called "totalExpenseAllowed" is used to keep the value of something between $4.75 and $8.20. The value adjustment works on values that are of type FLOAT, and binds them to the expense variable during the adjustment.

```
VALUE ADJUSTMENT totalExpenseAllowed (FLOAT expense)
IS
MIN ( MAX ( expense, 4.75), 8.20)
END VALUE ADJUSTMENT
```

The adjustment itself works in two steps. First, it uses the MAX function to make sure the value is not lower than $4.75. Then it uses the MIN function to make sure the value is not higher than $8.20. The adjustment itself is called with the ADJUST expression:

```
ADJUST ( FIRST (xs).expense, totalExpenseAllowed ))
```

found in the rate module 201. In this example, the ADJUST function uses the FIRST function to step through a sequence of values "xs", examine the expense attribute in each one, and then adjust its value with the totalExpenseAllowed value adjustment.

FIG. 4 and FIG. 5 illustrate basic steps in the processes of authoring, compiling, and running a program to implement a rating methodology in accordance with the present invention. The description of FIG. 4 and FIG. 5 are usefully understood with reference to the exemplary data structure shown in FIG. 6 used for the implementation of the present invention. In accordance with a particular implementation of the present invention, a user authors a rating methodology using a high level human readable programming language. Desirably, the high level language is a domain specific language that is readily implemented by a user familiar with the application domain, yet perhaps less familiar with programming techniques. An example of such a language is the PROBOL programming environment that is specific to insurance industry rating methodology implementations. This programming environment is described in greater detail in co-pending U.S. Patent Application Serial No. XX/XXX,XXX entitled "DATABASE PROGRAMMING

LANGUAGE" assigned to the assignee of the present invention and incorporated herein by reference.

As shown in FIG. 4, the authoring step 401 involves developing source code definitions of filters 302, modifiers 304 and value adjustments 306 in a constraint module 203. The source code is compiled in step 403 to class files such as JAVA class files or the equivalent used in the Java Programming environment. Class files maintain an object oriented structure and allow the constraint module to be executed on a variety of platforms by interpreting the class file at run time to platform specific code. In this manner, a class file can be created, stored, maintained, and upgraded without affecting behavior of any other class files.

In step 405 the class file(s) corresponding to a constraint module are associated with a geographic region as shown in FIG. 6 (e.g., a state, region, country, or market) in which the constraints are valid. Constraints may be hierarchically arranged such that more than one constraint module applies to a given region. For example, constraints imposed by a particular state's government (e.g., Colorado) will only be applicable to geographic areas within Colorado while constraints imposed by the Unites States Congress are applicable to every state. In such a case, when the geographic area is within Colorado both the constraint modules associated with Colorado and the United States will be applied. In step 407 the class files all generated constraint modules are stored within the database table for the geographic region associated with the constraint module. These are stored, for example, as binary large objects (BLOBs) shown in FIG. 6.

At runtime, a user initiates execution of a rate module 201 and specifies or selects a market in step 501. As used herein, a "market" is a geographic area or region that includes one or more geo-political regions. In other words, a market is defined by the relevant marketplace in which a particular product or products are offered and sold and not by arbitrary political boundaries. As a corporate customer may have employees that require insurance in a number of states, the market would include each of the states in which an insurable employee resides, for example. In step 503 the database table references are traversed to identify reference geo-political regions such as states and countries that are included in or implied by the specified market.

In step 505 the database table references are traversed to identify one or more constraint module(s) associated with the geo-political regions identified in step 503. The identified constraint modules are each associated with one or more of the class files stored in step 407 shown in FIG. 4. The identified class files are extracted from the database in step 507 (if they have not been previously extracted and cached) and attached to the active thread (i.e., the programatic thread in the main routine 201 that called the constraint module) in step 509. Where the class files have already been extracted from the database they are desirably cached in the file system following the initial use and so can ordinarily be used without further database access. The present invention can be implemented in a single threaded environment but advantageously is implemented in a multithreaded environment where multiple threads of execution run concurrently and each thread can be associated with or more constraint modules by attaching the appropriate class files in step 509.

As the rate module 201 executes, predefined statements (represented by code segments in a compiled rate module 201) will make calls to constraints. By way of example, a FILTER statement calls a filter constraint, an ADJUST statement calls and adjust constraint 306 and a SAVE statement calls a modify constraint thereby applying the modify constraints automatically upon saving computed rates to database 305 (shown in FIG. 3).

Each call to a constraint is configured to include one or more arguments that are passed to the attached constraint code. As the rate module 201 makes calls to constraints (i.e., filters, adjustments, or modifiers) the set of constraints attached to the thread are examined in step 511 to determine which if any are applicable to the arguments of the calling statement.

In a specific example, rate modules 201 "activate" constraint modules implicitly as they are used. An important feature of the present invention is that constraint modules need not be explicitly referenced by name by rate modules so that the set of active constraint modules may change without the rate module requiring updates. For the following example, assume that two constraint modules are defined as:

```
CONSTRAINT MODULE stateConstraints
DISPLAYED AS "My Favorite State Constraints"
IS ...
END MODULE
```

Constraint Module 1

```
CONSTRAINT MODULE marketConstraints
DISPLAYED AS "Top 10 Market Constraints"
IS ...
END MODULE
```

Constraint Module 2

When a rate module is running the constraint modules required by the rate module are made active (i.e., attached to an active thread). In other words, the filters, modifiers, and value adjustments defined in those constraint modules can be applied in the rate module. The calling expression includes arguments that define a particular set of database class instances to which the filter is to be applied. As describe hereinbefore, any database class may be associated with multiple constraints. Accordingly, in response to the calling expression the rating system in accordance with the present invention calls all the active constraints that are associated with the database class of those instances specified in the calling expression.

In a preferred implementation, constraint modules can be conditionally applied by including an "application clause" in the rate module 201. The application clause modifies the calling expression to determine whether the constraint should be applied at all even thought it is already attached. The application clause defines certain conditions that must be satisfied before the constraint is applied to the database class instances. In this manner, the rate module 201 can be given more control over the application of constraints. The application clause can be omitted, however, so that the active constraints are applied to all instances of the associated database class.

Although the invention has been described and illustrated with a certain degree of particularity, it is understood that the present disclosure has been made only by way of example, and that numerous changes in the combination and arrangement of parts can be resorted to by those skilled in the art without departing from the spirit and scope of the invention, as hereinafter claimed.

WE CLAIM:

1. A computer implemented rating methodology in a plurality of externally defined contexts, the method comprising the steps of:

providing a database comprising records describing objects to be rated;

creating at least one context-specific constraint module that is selectively enabled to constrain rating of selected records; and

providing a context-generic rating module comprising programming constructs that access records from the database, perform basic computations on the records to generate output data, and selectively enable the constraint module.

2. The computer implemented method of claim 1 wherein the constraint module comprises programming constructs that implement a filter to selectively exclude records from the rating module.

3. The computer implemented method of claim 1 wherein the constraint module comprises programming constructs that implement value adjustments that modify the basic computations performed by the rating module.

4. The computer implemented method of claim 1 wherein the constraint module comprises programming constructs modify the output data generated by the rating module.

5. The computer implemented methods of claim 1 further comprising the steps of:

altering the rating methodology by modifying the constraint module without modification to the rating module.

6. The computer implemented methods of claim 1 further comprising selecting the at least one context-specific constraint module to be enabled based upon a current one of the externally defined contexts.

7. A method for rating a plurality of stored objects to generate a set of context-specific rate quotes comprising the step of:

defining a set of context-generic operations configured to operate on the objects to generate a set of rate outputs;

defining a plurality of context-specific constraint modules where each constraint module is configured to constrain the context-generic operations;

accessing objects from the database;

executing the context-generic operations on the accessed objects to generate the set of rate outputs; and

enabling at least one of the context-specific constraint modules to modify the set of rate outputs, wherein the modified set of rate outputs forms the context-specific rate quotes.

8. The method of claim 7 wherein the step of enabling the constraint module further comprises filtering the accessed objects to selectively exclude objects from the executing step.

9. The method of claim 7 wherein the step of enabling the constraint module further comprises adjusting values that modify the computations performed during the executing step.

10. The method of claim 7 further comprising the step of: writing the context specific rate quotes to the database, wherein the step of enabling the constraint module further comprises modifying the rate outputs after they are generated during the execution step before the step of writing the rate outputs to the database.

11. The method of claim 7 wherein the step of executing the context-generic operations comprise executing context-generic programming statements that describe the operations on a local computer.

12. The method of claim 11 further comprising the step of storing the context-specific constraint modules on a remote computer system and the step of executing further comprises selectively accessing the context-specific constraint modules from the remote computer system as needed.

13. The method of claim 7 further comprising the step of altering the rating methodology by modifying the constraint module definition without modification to the set of context-generic operations.

14. A computer system for implementing an rating methodology to generate rate quotes for a set of data objects, the computer system comprising:

a first computer having a processor and memory;

a database coupled to the first computer comprising records describing the set of data objects to be rated;

a plurality of context-specific constraint modules that are selectively enabled to constrain rating of selected data objects; and

a context-generic rating module comprising programming constructs that access records from the database, perform basic computations on the records to generate output data, and selectively enable the constraint module.

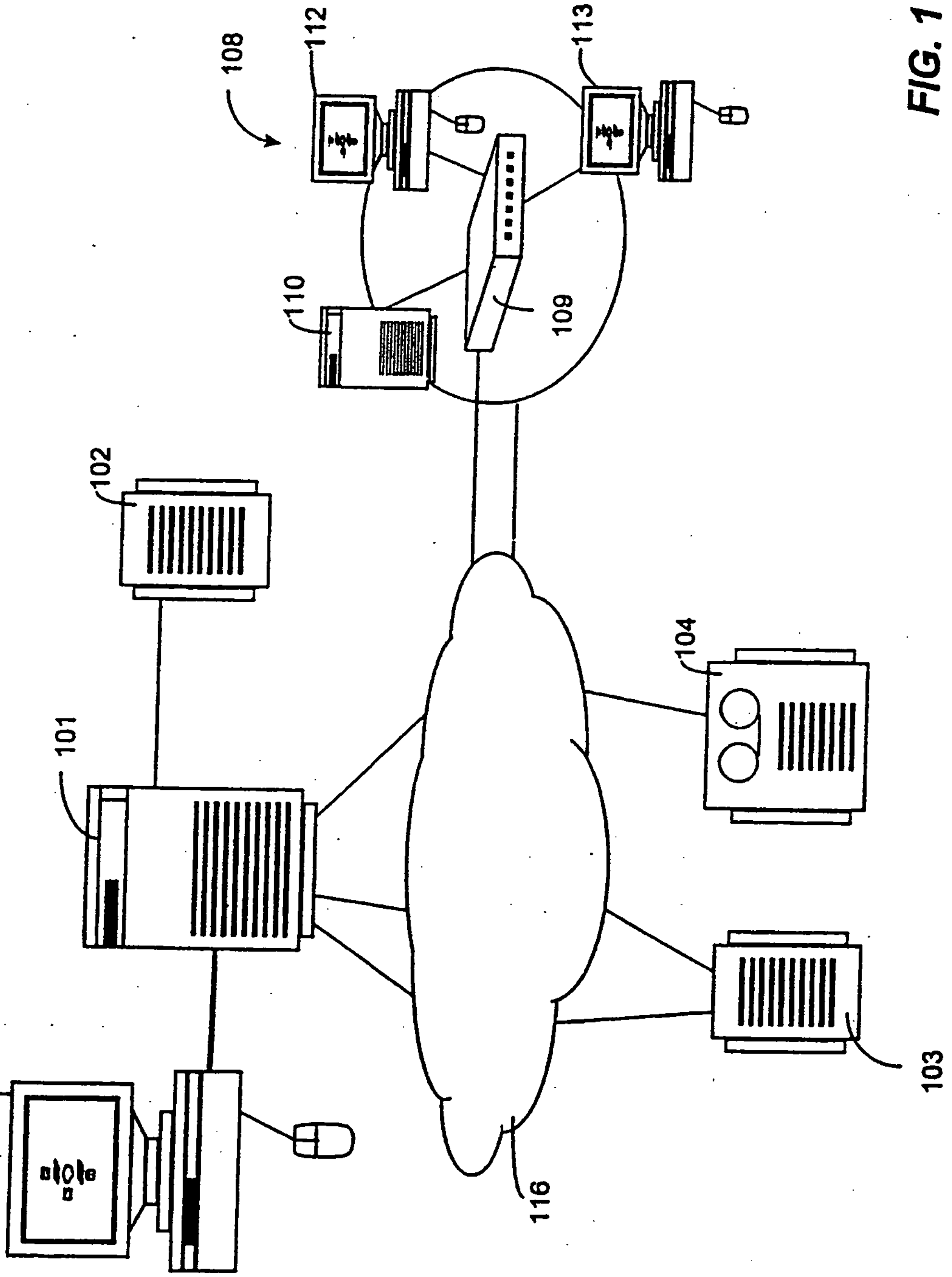


FIG. 1

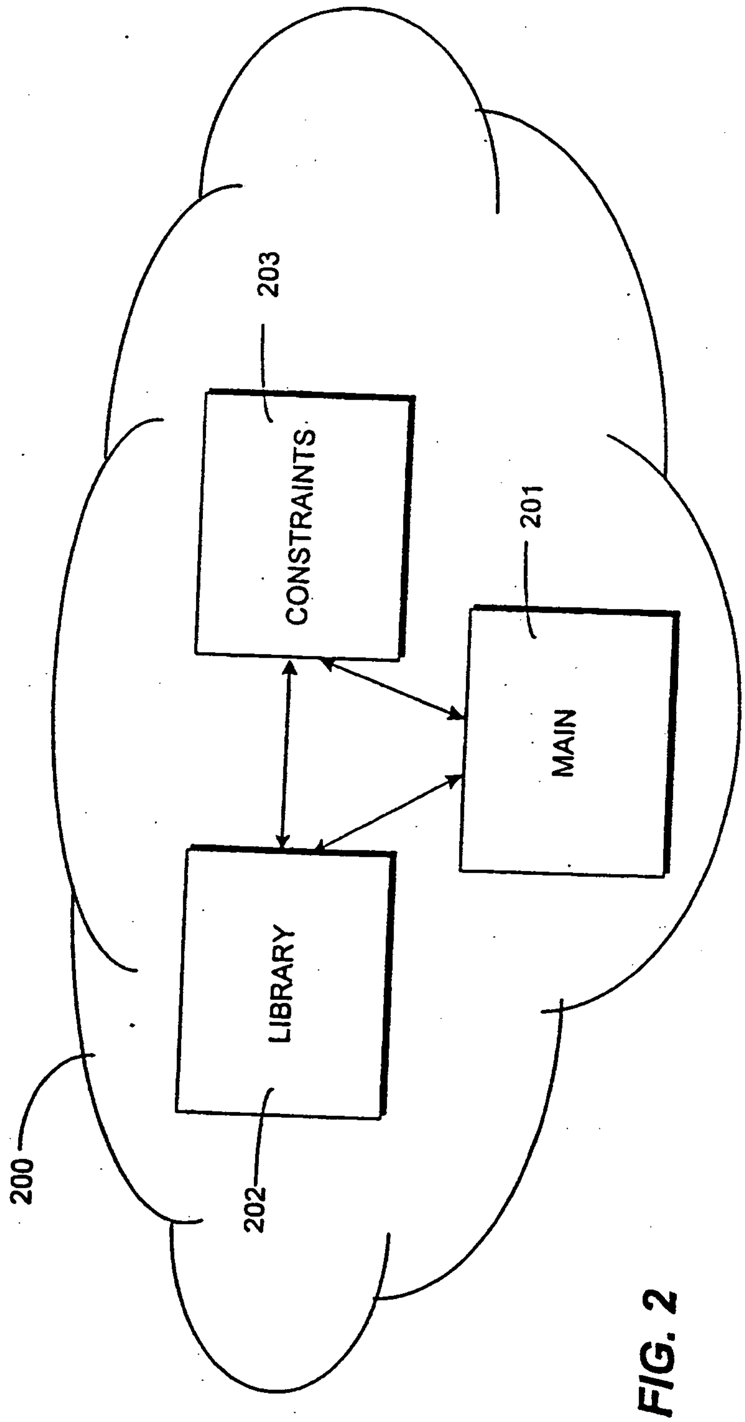


FIG. 2

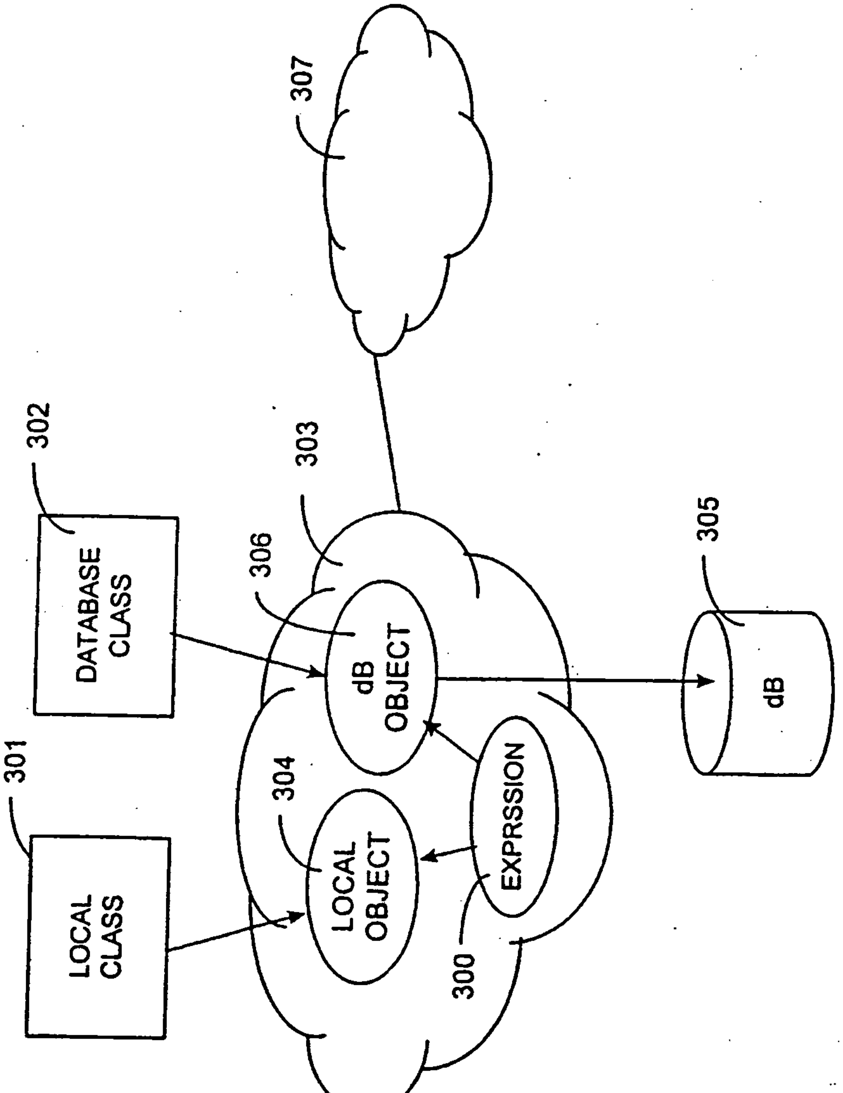

307
302
DATABASE CLASS
303
306
dB OBJECT
305
dB
301
LOCAL CLASS
304
LOCAL OBJECT
EXPRSSION
300

FIG. 3

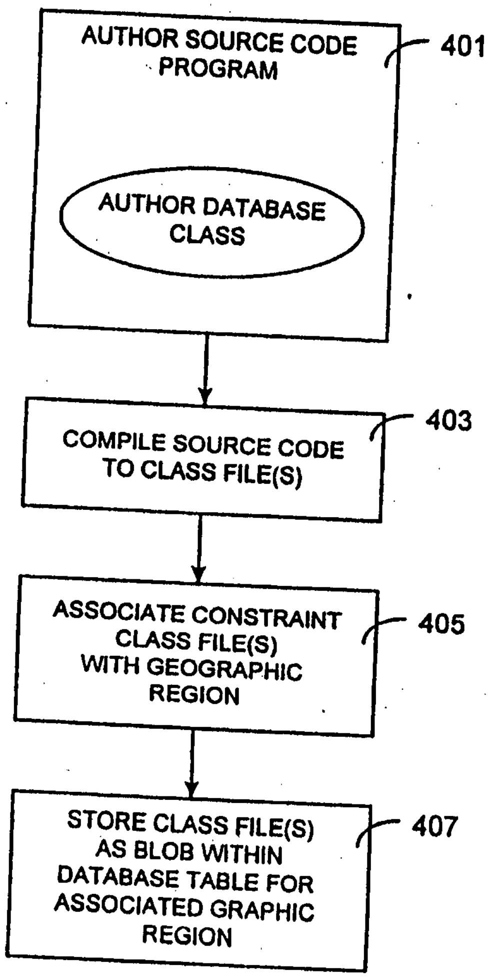

AUTHOR SOURCE CODE PROGRAM
AUTHOR DATABASE CLASS
401
COMPILE SOURCE CODE TO CLASS FILE(S)
403
ASSOCIATE CONSTRAINT CLASS FILE(S) WITH GEOGRAPHIC REGION
405
STORE CLASS FILE(S) AS BLOB WITHIN DATABASE TABLE FOR ASSOCIATED GRAPHIC REGION
407

FIG. 4

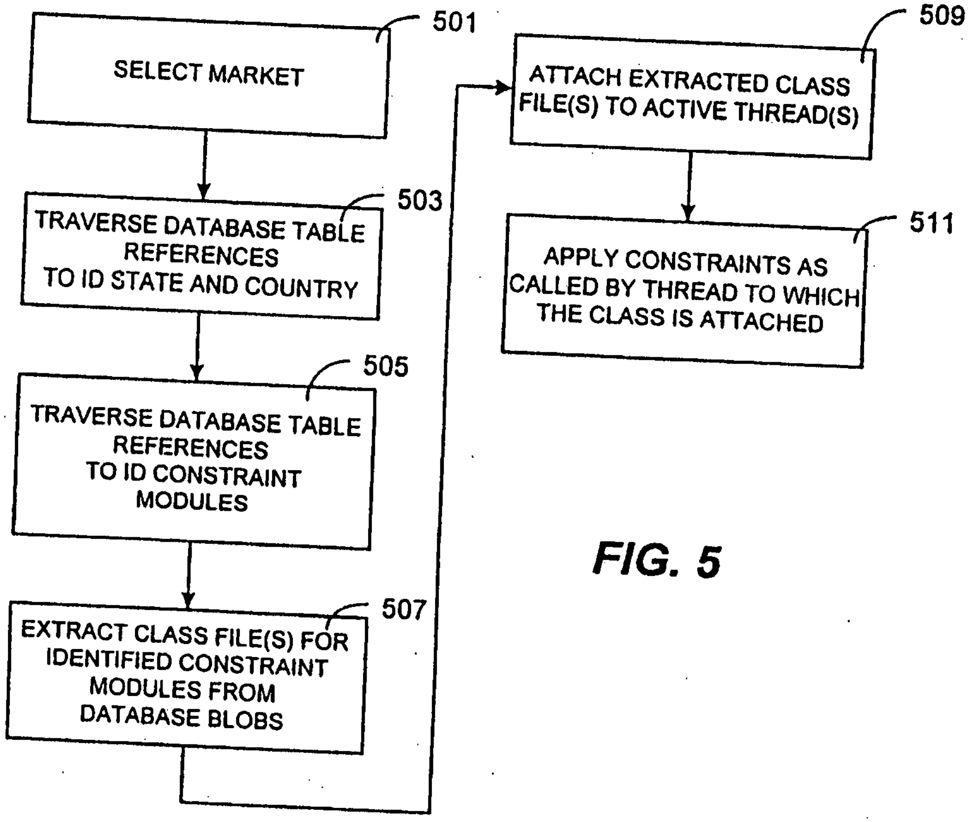

SELECT MARKET
501
TRAVERSE DATABASE TABLE REFERENCES TO ID STATE AND COUNTRY
503
TRAVERSE DATABASE TABLE REFERENCES TO ID CONSTRAINT MODULES
505
EXTRACT CLASS FILE(S) FOR IDENTIFIED CONSTRAINT MODULES FROM DATABASE BLOBS
507
ATTACH EXTRACTED CLASS FILE(S) TO ACTIVE THREAD(S)
509
APPLY CONSTRAINTS AS CALLED BY THREAD TO WHICH THE CLASS IS ATTACHED
511

FIG. 5

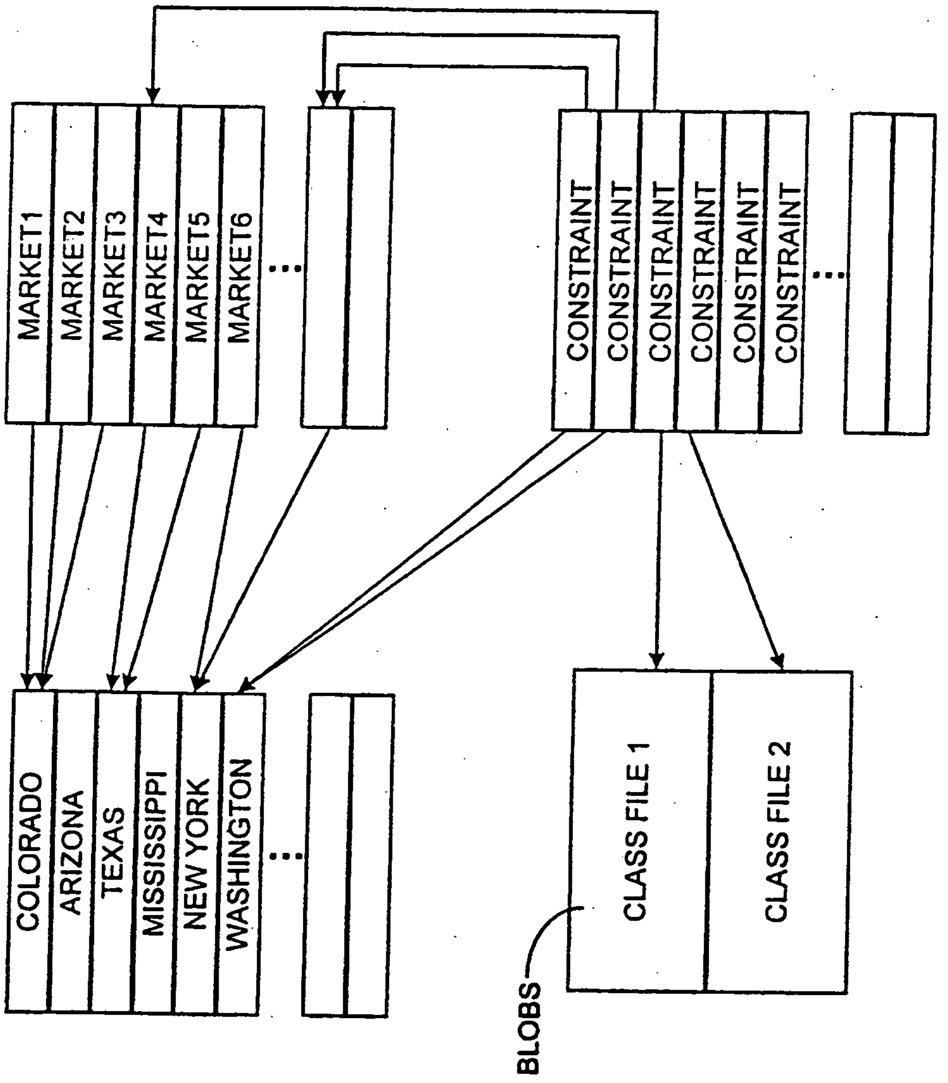

MARKET1
MARKET2
MARKET3
MARKET4
MARKET5
MARKET6
CONSTRAINT
CONSTRAINT
CONSTRAINT
CONSTRAINT
CONSTRAINT
CONSTRAINT
COLORADO
ARIZONA
TEXAS
MISSISSIPPI
NEW YORK
WASHINGTON
CLASS FILE 1
CLASS FILE 2
BLOBS

FIG. 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/00737

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :G06F 15/21, 17/30, 19/00,
US CL :705/1, 2, 4, 20, 277/2,202, 364/401
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : G06F 15/21, 17/30, 19/00,

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS
RATING, PRICING, COMPUTING, FILTERING, GENERIC, GENERAL, MODIFYING

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 4,831,526 A(LUCHS et al) 16 May 1989, entire document.	1-14
X	US 5,655,085 A (RYAN et al) 05 August 1997, Figures 1-29B	1-14
Y,P	US 5,907,848 A (ZAIKEN et al) 25 May 1999, enitire document.	1-14
Y,P	US 5,870,733 A (BASS et al) 09 February 1999, entire document.	1-14
Y,P	US 5,930,760 A (ANDERTON et al) 27 July 1999, entire document.	1-14
A	US 5,664,119 A (Johnson et al) 02 September 1997, entire document.	1-14

□ Further documents are listed in the continuation of Box C. □ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance		
"E"	earlier document published on or after the international filing date	"X"	document of particular relevance, the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Y"	document of particular relevance, the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed	"&"	document member of the same patent family

Date of the actual completion of the international search 16 MARCH 2000	Date of mailing of the international search report 18 APR 2000
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer TODD VOELT James R. Matthews Telephone No. (703) 308-0000

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/32342

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,523,942 A (TYLER, et al.) 04 June 1996, col. 6, line 10.	1, 27, 30, 38, 46, 53, 69, 79, 83
Y	US 5,655,085 A (RYAN et al.) 05 August 1997, col. 5, line 66.	1, 27, 30, 38, 46, 53, 69, 79, 83
Y	US 4,975,840 A (DeTORE et al.) 04 December 1990, col. 5, line 4.	1, 27, 30, 38, 46, 53, 69, 79, 83
Y	TAUHERT, CHRISTY. "ITSA, ISI Alliance Promotes Point-of-Sale Underwriting," Insurance and Technology, 01 December 1996, pp. 1-2.	1-27, 30, 38, 46, 53, 69, 79, 83
Y	DUNLOP, NEIL. "Famous Last Words," Canadian Underwriter, 01 September 1997, pp. 1-3.	1-100
Y	MORNINGSTAR, MATTHEW. "Drifting Toward Cyberspace," Best's Review. 01 May 1998, pp. 1-4.	1-100
Y	MENCHEN, JOHN. "Insurance Pioneers Blaze Trail on World Wide Web," National Underwriter, 16 September 1996, pp. 1-3.	1-27, 30, 38, 46, 53, 69. 79. 83
A	LONGO, TRACEY. "Insurance Online: Faster, Not Cheaper," Kiplinger's Personal Finance Magazine, 01 October 1996, p.1.	1-100